Validation and exception handling is what we use to return predictable results.

We’ll look at 3 method results:

* Return a value
* Return an exception
* Return multiple values.
* Returning null

Garbage in/garbage out. We want to create the situation where our methods can deal with garbage in by returning a predictable value.

A clean method shouldn’t return garbage no matter what is passed in.

If garbage is passed in the method must notify the calling code somehow.

Returning a bool doesn’t tell the caller what went wrong.

Instead of attempting to return a value when invalid data or an issue occurs, consider throwing an exception.

Exceptions were originally designed to report execution failures and not for validating business rules. Is this appropriate?

The alternative is to return multiple values which makes sense when you’re expecting some value and an execution result.

Returning null in place of an expected value could be an indication something went wrong but doesn’t provide any details on the problem. The thing to consider what null to use.

Nullable types – allow an integer or decimal to contain either the value or a null. Some developers recommend never returning null, others think null is appropriate when an object is expected.

The downside of returning a value is the calling code needs to check the result. This means the calling code might forget and carry on with invalid values

Exceptions aren’t meant to handle validation, they’re meant to deal with exceptional conditions and failure reporting.

If returning a single value isn’t enough and throwing an exception isn’t quite right, especially with validation cases, what should we do? How can you return a value and more detailed messages?

You can return multiple values from a method. This isn’t always appropriate but there are times when it is, for example validation.

Ways to return multiple values from a method:

* Ref params. In general they’re not recommended because they are not intuitive.
* Out params. Also not recommended.
* Tuple. A little clumsy as you can only work with Item1, Item2…etc. Not intuitive, what it’s returning
* Objects. The preferred way to return multiple values. A class just for return values. One practice is to define an *OperationResult* class. Standard definition is a bool and a list of messages to pass back.

Tuple.Create(…) – creates a new tuple

Null reference – keyword in C# that doesn’t refer to any object.

Strings can return null

Value types can return null like this:

Nullable<decimal> result = null

The Nullable type above is a generic struct that contains the value and a null value.

An object can return null. The default of any reference type is null.

Collections can be null.

2 schools of thought on returning nulls:

1. Don’t do it. Requires the calling code to always check. If one is missed will cause a null exception.
2. Return a null everytime the result is actually nothing.

Null object pattern - a design pattern that defines an object with defined neutral or null behaviour.